

FruitKiwi Technical Handover

This guide explains the transferred application package, the buyer-owned services needed to operate it, the deployment sequence, and the checks to complete before public launch.

1. Product and system overview

FruitKiwi is a browser-based B2B marketplace for specialty produce sourcing. It includes public discovery pages, buyer and supplier workspaces, a separate owner console, a realtime application backend, file storage, scheduled maintenance tasks, content management, audit records, and billing for supplier plans and promoted listings.

Produce payments are handled directly between buyers and suppliers. Stripe is used only for FruitKiwi software plans and paid visibility.

2. Main application areas and user roles

- **Public area:** produce search, supplier directory, storefronts, pricing, guides, contact, privacy, and terms pages.
- **Buyer area:** onboarding, saved items, buying lists, requests for quotes, quote comparison, messages, notifications, and sourced-only reviews.
- **Supplier area:** approval submission, storefront editing, catalogue management, seasonal availability, request inbox, quote creation, lead analytics, billing, and invoices.
- **Owner area:** user management, supplier approvals, listing moderation, requests, reports, plans, promotions, taxonomy, content, exports, privacy requests, maintenance controls, and audit history.

Public visitors, authenticated buyers, authenticated suppliers, and platform owners have separate permissions. Privileged checks run in the backend, not in editable browser profile data.

3. Buyer-safe architecture summary

- **Frontend:** React, TypeScript, Vite, React Router, Tailwind CSS, and static production output.
- **Authentication:** Clerk sign-in with a Convex JWT template named `convex`.
- **Backend:** Convex functions, database, realtime updates, scheduled jobs, file storage, rate limits, content management, and audit records.
- **Billing:** Stripe Checkout, customer portal, invoices, subscriptions, promotions, and verified webhooks.
- **Hosting:** any HTTPS host that can serve the generated static files and preserve the included single-page route fallback.
- **Package manager:** `pnpm`.

The browser receives public configuration only. Authentication secrets, Stripe secret keys, webhook secrets, and seed secrets must remain in the backend deployment environment.

4. Project structure

- `src/` contains the browser application, pages, shared components, product defaults, and styles.
- `src/home/` contains the public homepage.

- `src/pages/` contains authentication and owner console entry pages.
- `convex/` contains the data model, queries, mutations, actions, scheduled jobs, authentication checks, billing, storage, content, reports, and seed tools.
- `convex/lib/` contains shared authorization, rate-limit, and upload helpers.
- `public/` contains static brand assets, produce images, search files, and the hosting route fallback.
- `.env.example` documents the required public and server-side variable names.
- `srs.md` describes the intended product scope.
- `SRS-COMPLIANCE.md` maps the implemented product areas to that scope.

5. Required buyer-owned services and configuration

Create fresh buyer-owned accounts. Do not reuse seller credentials. Use obvious placeholders until each service is ready.

Public browser environment

Create `.env.local` from `.env.example` and set only public values:

```
VITE_CONVEX_URL=https://YOUR_DEPLOYMENT.convex.cloud
VITE_CLERK_PUBLISHABLE_KEY=YOUR_PUBLIC_CLERK_KEY
VITE_STRIPE_PUBLISHABLE_KEY=YOUR_PUBLIC_STRIPE_KEY
VITE_SITE_URL=https://YOUR_SITE_URL
```

Never place a secret key in a variable beginning with `VITE_`. Browser visitors can read those values.

Backend deployment environment

Set these in the buyer's Convex deployment, not in browser files:

```
CLERK_SIGNING_KEY=https://YOUR_CLERK_ISSUER
STRIPE_API_KEY=YOUR_SERVER_STRIPE_KEY
STRIPE_WEBHOOK_SECRET=YOUR_WEBHOOK_SIGNING_SECRET
STRIPE_SITE_ID=YOUR_DEPLOYMENT_NAME
SITE_URL=https://YOUR_SITE_URL
APP_ID=fruitkiwi
APP_NAME=FruitKiwi
SEED_SECRET=YOUR_LONG_RANDOM_SEED_SECRET
```

Optional allowlists can be configured for controlled buyer or supplier accounts. Use buyer-owned user IDs and emails only. Keep every server value out of screenshots, documentation, and source control.

6. Deployment sequence

1. Open the transferred project folder and review `.env.example`.
2. Create the buyer-owned Clerk application, Convex deployment, Stripe account configuration, domain DNS, and HTTPS hosting.
3. In Clerk, create the Convex JWT template named `convex`. Set the matching Clerk issuer in the Convex authentication configuration and deployment environment.
4. Add the public values to `.env.local`. Add server-only values to the Convex deployment environment.
5. Install the locked dependencies with `pnpm install`.

6. Connect to the intended development deployment with `pnpm exec convex dev` and confirm that functions compile.

7. Run the protected demo seed only against the intended environment:

```
pnpm exec convex run seed:ensure '{"secret":"YOUR_SEED_SECRET"}'
```

8. Run `pnpm test`, then run `pnpm build` once for the release candidate.

9. Deploy the Convex functions to the buyer's production deployment using the standard Convex production workflow.

10. Build with the production public values and upload the contents of `dist/` to the hosting document root. Include the generated route fallback file.

11. Point the domain to the host, enable HTTPS, and test direct visits to nested routes.

Run seed actions carefully. Do not seed or reset an environment that contains real marketplace data.

7. Payments, authentication, email, storage, and smart tools

Stripe billing: Stripe covers supplier subscriptions and promoted placements only. Configure the required products, prices, customer portal, and webhook route in the buyer's account. The webhook flow must preserve the application and deployment routing metadata and verify the original Stripe signature before changing access.

Required subscription events are:

- `checkout.session.completed`
- `customer.subscription.updated`
- `customer.subscription.deleted`

The webhook handler stores events once and grants or removes paid access from verified billing status. Test successful checkout, renewal state, cancellation, failed payment handling, customer portal access, and promoted placement expiry before launch.

Clerk authentication: Use one buyer-owned Clerk application for the browser key and Convex JWT issuer. Verify sign-in, sign-out, role onboarding, session refresh, and protected function access. Owner access must be controlled by the secure backend allowlist and the separate owner console controls.

Email and contact handling: The current product does not include a transactional email provider or SMS service. In-app notifications and stored contact requests are available. If email delivery is needed, the buyer can add a provider later and must keep its secret key in the backend environment.

File storage: Supplier images, listing images, message attachments, and related uploads use Convex storage. The backend validates ownership, file type, and size before issuing upload access. Confirm upload limits and retention rules before accepting public content.

Smart sourcing tools: The matching score, request completeness checks, seasonal warnings, quote comparison, supplier quality score, and suggested alternatives use deterministic rules. No external AI service or AI key is required.

8. Security and credential rotation

- Replace every seller-owned public key, secret key, webhook secret, seed secret, administrator credential, and service account.

- Deliver initial owner access through a secure channel, then change it immediately.
- Keep server secrets in the Convex deployment environment only.
- Confirm backend ownership checks for buyer, supplier, and owner actions.
- Review upload validation, message and request rate limits, webhook signature checks, event idempotency, audit records, and account suspension flows.
- Remove demo accounts and data that should not appear publicly.
- Review legal pages, privacy requests, data export, account deletion, cookie use, and local marketplace obligations.
- Never run the protected seed against live customer data.

9. Functional verification checklist

Public and buyer flow

- Public produce and supplier pages show only approved, published records.
- Search filters and direct page links work on desktop and mobile.
- A buyer can finish onboarding, save items, create a multi-line request, invite suppliers, compare quotes, send messages, mark a request sourced, and leave a review.

Supplier flow

- A supplier can create a storefront, submit it, receive owner approval, publish catalogue items, view matched requests, send a quote, exchange messages, and view lead activity.
- Free and paid plan limits are enforced without deleting supplier data after downgrade.
- Catalogue CSV import is available only to the intended paid plan.

Owner and billing flow

- The owner can manage users, suppliers, listings, plans, promotions, content, taxonomy, reports, privacy requests, and maintenance mode.
- Stripe test checkout grants the correct plan or promotion only after a verified event.
- Duplicate webhook delivery does not grant access twice.
- Cancellation or failed status removes paid access without deleting catalogue records.

Release checks

- `pnpm test` completes successfully.
- `pnpm build` creates the static release package.
- Nested route refreshes return the application instead of a server 404 page.
- No secret appears in browser source, build output, logs, or public documentation.

10. Backup, monitoring, and maintenance

- Keep an encrypted copy of the transferred source package and release artifacts.
- Schedule regular Convex data exports and document the restore procedure.

- Monitor Convex function errors, storage use, scheduled jobs, and rate-limit activity.
- Monitor Clerk sign-in errors and suspicious account activity.
- Monitor Stripe webhook delivery, subscription status, failed payments, disputes, and customer portal access.
- Renew the domain, HTTPS certificate, hosting plan, and third-party services before expiry.
- Review dependencies and security notices on a planned schedule. Test changes in a separate environment before production.
- Keep an operations log for plan changes, moderation actions, incidents, backups, and releases.

11. Product and support boundaries

FruitKiwi manages discovery, supplier storefronts, seasonal catalogues, requests, quotes, comparison, messages, sourcing intent, reviews, supplier software plans, and paid visibility.

The application does not include produce checkout, escrow, wallets, supplier payouts, Stripe Connect, warehouse inventory, tax calculation, shipping integrations, map or geocoding services, accounting, SMS, scraping, external AI APIs, or native mobile apps. Inventory is indicative marketplace availability, not warehouse stock control.

No live revenue, traffic, customer base, uptime history, or production payment account is represented as part of the starter asset. The buyer must complete deployment, third-party setup, legal review, supplier onboarding, content review, and go-to-market work.

Codeblix provides up to 30 days of reasonable technical handover support. This covers transfer and initial setup questions. It does not include third-party fees, ongoing operations, legal advice, supplier recruitment, marketing, unlimited custom development, or guaranteed results.